An Introduction to Sequential Rule Mining

Philippe Fournier-Viger
http://www.philippe-Fournier-viger.com

Source code and datasets available in the SPMF library
Introduction

• More and more data!
• A need to analyze data to find interesting patterns
• **Pattern mining**: using algorithms to find interesting patterns in data.
• An important type of data is **sequences**.
• Today, we will discuss how to analyze sequences to find **sequential rules**.
What is a **discrete sequence**?

**Sequence**: an ordered list of symbols

**Sequence of purchases**

- Computer
- Monitor
- Router

**Sequence of words**

Where → are → you → going?
What is a discrete sequence?

Sequence: an ordered list of symbols

Sequences of webpage clicks

Webpage A → Webpage B → Webpage C → ...

Sequences of activities

Home → Watching movies → Visit museum → ...

Home
Definition: Items

Let there be a set of items (symbols) called $I$.

Example: $I = \{a, b, c, d, e, f, g\}$

- $a = \text{apple}$
- $b = \text{bread}$
- $c = \text{cake}$
- $d = \text{dattes}$
- $e = \text{eggs}$
- $f = \text{fish}$
- $g = \text{grapes}$
Definition: **Itemset**

An itemset is a set of **items** that is a subset of \( I \).

**Example:** \( \{a, b, c\} \) is an itemset containing 3 items.

\( \{d, e\} \) is an itemset containing 2 items.

- Note: an itemset cannot contain a same item twice.
- An itemset having \( k \) items is called a **\( k \)-itemset.**
Definition: Sequence

A discrete sequence $S$ is a an ordered list of itemsets $S = \langle X_1, X_2, \ldots, X_n \rangle$ where $X_j \subseteq I$ for any $j \in \{1,2\ldots n\}$

Example 1: $\langle \{a, b\}, \{c\} \rangle$ is a sequence containing two itemsets.

It means that a customer purchased apple and bread at the same time and then purchased cake.

Example 2: $\langle \{a\}, \{a\}, \{c\} \rangle$
A **sequence database** is one or more sequences.

<table>
<thead>
<tr>
<th>SID</th>
<th>sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>&lt;{a}, {a,b,c} {a, c} {d} {c, f}&gt;</td>
</tr>
<tr>
<td>2</td>
<td>&lt;{a, d}, {c} {b, c} {a, e}&gt;</td>
</tr>
<tr>
<td>3</td>
<td>&lt;{e, f}, {a, b} {d, f} {c}, {b}&gt;</td>
</tr>
<tr>
<td>4</td>
<td>&lt;{e}, {g}, {a, f} {c} {b}, {c}&gt;</td>
</tr>
</tbody>
</table>

Here we have four sequences.

Each sequence has a unique sequence identifier (SID).
Sequential pattern mining

It is a popular data mining task, where the goal is to find sequential patterns.

<table>
<thead>
<tr>
<th>SID</th>
<th>sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>&lt;{a}, {a,b,c} {a, c} {d} {c, f}&gt;</td>
</tr>
<tr>
<td>2</td>
<td>&lt;{a, d}, {c} {b, c} {a, e}&gt;</td>
</tr>
<tr>
<td>3</td>
<td>&lt;{e, f}, {a, b} {d, f} {c}, {b}&gt;</td>
</tr>
<tr>
<td>4</td>
<td>&lt;{e}, {g}, {a, f} {c} {b}, {c}&gt;</td>
</tr>
</tbody>
</table>
Sequential pattern mining

**Sequential pattern**: a subsequence that appear in many sequences of a sequence database

<table>
<thead>
<tr>
<th>SID</th>
<th>sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>&lt;{a}, {a,b,c} {a, c} {d} {c, f}&gt;</td>
</tr>
<tr>
<td>2</td>
<td>&lt;{a, d}, {c} {b, c} {a, e}&gt;</td>
</tr>
<tr>
<td>3</td>
<td>&lt;{e, f}, {a, b} {d, f} {c}, {b}&gt;</td>
</tr>
<tr>
<td>4</td>
<td>&lt;{e}, {g}, {a, f} {c} {b}, {c}&gt;</td>
</tr>
</tbody>
</table>

<{a},{f}> is a **sequential pattern**
Sequential pattern mining

**Sequential pattern**: a subsequence that appear in many sequences of a sequence database

<table>
<thead>
<tr>
<th>SID</th>
<th>sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>&lt;a&gt;, {a,b,c} {a, c} {d} {c, f}&gt;</td>
</tr>
<tr>
<td>2</td>
<td>&lt;a, d}, {c} {b, c} {a, e}&gt;</td>
</tr>
<tr>
<td>3</td>
<td>&lt;e, f}, {a, b} {d, f} {c}, {b}&gt;</td>
</tr>
</tbody>
</table>
| 4   | <e}, {g}, {a, f} {c} {b}, {c}>

<{a},{f}> is a **sequential pattern**

Its **support** is 50% (it appears in 50% of the sequences).
Sequential pattern mining

Input:
- A sequence database (a set of sequences)
- A \textit{minsup} threshold

Output:
- All subsequences having a support greater or equal to \textit{minsup}.

Example: \textit{minsup} = 50 \% (2 sequences)

<table>
<thead>
<tr>
<th>IFD</th>
<th>sequence</th>
<th>Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>{a}, {a,b,c}, {a, c}, {d}, {c, f}</td>
<td>100 %</td>
</tr>
<tr>
<td>2</td>
<td>{a, d}, {c}, {b, c}, {a, e}</td>
<td>50 %</td>
</tr>
<tr>
<td>3</td>
<td>{e, f}, {a, b}, {d, f}, {c}, {b}</td>
<td>50 %</td>
</tr>
<tr>
<td>4</td>
<td>{e}, {g}, {a, f}, {c}, {b}, {c}</td>
<td>...</td>
</tr>
</tbody>
</table>

Some popular algorithms


They all have the same input and output. The difference is performance due to optimizations, search strategies and data structures!

**Fast implementations** available in the [SPMF library](http://www.philippe-fournier-viger.com/spmf/).
But there is a problem...

Let look at the pattern \(<\{a\},\{f\}>\)
We might think that if someone buys « a », he will buy « f » afterward.

<table>
<thead>
<tr>
<th>SID</th>
<th>sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(&lt;{a}, {a,b,c} {a, c} {d} {c, f}&gt;)</td>
</tr>
<tr>
<td>2</td>
<td>(&lt;{a, d}, {c} {b, c} {a, e}&gt;)</td>
</tr>
<tr>
<td>3</td>
<td>(&lt;{e, f}, {a, b} {d, f} {c}, {b}&gt;)</td>
</tr>
<tr>
<td>4</td>
<td>(&lt;{e}, {g}, {a, f} {c} {b}, {c}&gt;)</td>
</tr>
</tbody>
</table>
But there is a problem...

Let look at the pattern `<{a},{f}>`
We might think that if someone buys « a », he will buy « f » afterward.

No! Only **50% of the time**!

<table>
<thead>
<tr>
<th>SID</th>
<th>sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><code>&lt;{a}, {a,b,c} {a, c} {d} {c, f}&gt;</code></td>
</tr>
<tr>
<td>2</td>
<td><code>&lt;{a, d}, {c} {b, c} {a, e}&gt;</code></td>
</tr>
<tr>
<td>3</td>
<td><code>&lt;{e, f}, {a, b} {d, f} {c}, {b}&gt;</code></td>
</tr>
<tr>
<td>4</td>
<td><code>&lt;{e}, {g}, {a, f} {c} {b}, {c}&gt;</code></td>
</tr>
</tbody>
</table>

Thus, sequential patterns can be **misleading**!
How to address this problem?

• We would like to find patterns that have the form of rules.
• We want to measure the confidence (probability) that some item(s) will follow some other item(s).
• **Solution**: finding **sequential rules**

Restaurant ➔ 75% ➔ Watching movies
Two main types of sequential rules

1) Standard Sequential rules
2) Partially-ordered Sequential rules
1) Standard Sequential rules

**Standard Sequential rules**: Rules of the form $X \rightarrow Y$, where $X$ and $Y$ are sequential patterns.

**Example**: $\langle\{a\}, \{b,c\}\rangle \rightarrow \langle\{d\}, \{e\}\rangle$
1) Standard Sequential rules

**Standard Sequential rules**: Rules of the form $X \rightarrow Y$, where $X$ and $Y$ are sequential patterns.

**Example**: $\langle \{a\}, \{b,c\} \rangle \rightarrow \langle \{d\}, \{e\} \rangle$

- Several algorithms to find this type of rules such as **RuleGen** (Zaki, 2001).
- **Main idea**: find sequential patterns and then combine them to make rules.
1) Standard Sequential rules

**Standard Sequential rules**: Rules of the form $X \rightarrow Y$, where $X$ and $Y$ are sequential patterns.

**Example**: $<\{a\}, \{b,c\}> \rightarrow <\{d\}, \{e\}>$

- Two thresholds must be set by the user:
  - minimum support $> 0$
  - minimum confidence $> 0$
- **Support**: how many sequences contain a rule
- **Confidence**: how many sequences contain a rule divided by how many sequences contain its antecedent
1) Standard Sequential rules

Example: \(<\{a\}, \{b\}\> \rightarrow \langle f\rangle\)

Support: 1 sequences (25%)
Confidence: \(1 / 4 = 0.25\) (25%)

<table>
<thead>
<tr>
<th>SID</th>
<th>sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(&lt;{a}, {a,b,c} {a, c} {d} {c, f}&gt;</td>
</tr>
<tr>
<td>2</td>
<td>(&lt;{a, d}, {c} {b, c} {a, e}&gt;</td>
</tr>
<tr>
<td>3</td>
<td>(&lt;{e, f}, {a, b} {d, f} {c}, {b}&gt;</td>
</tr>
<tr>
<td>4</td>
<td>(&lt;{e}, {g}, {a, f} {c} {b}, {c}&gt;</td>
</tr>
</tbody>
</table>
But there is a problem...

We may find some sequential rules that are very similar but have only some **small ordering variations**.

For example:

<table>
<thead>
<tr>
<th>Rule</th>
<th>Support</th>
<th>Confidence</th>
</tr>
</thead>
<tbody>
<tr>
<td>(&lt;{a}, {b}&gt; \rightarrow {f}&gt; )</td>
<td>25%</td>
<td>25%</td>
</tr>
<tr>
<td>(&lt;{b}, {a}&gt; \rightarrow {f}&gt; )</td>
<td>25%</td>
<td>50%</td>
</tr>
</tbody>
</table>

These rules may actually represent the same situation!
2) Partially-Ordered Sequential rules

**Partially-Ordered Sequential rules**: Rules of the form $X \rightarrow Y$, where $X$ and $Y$ are itemsets that are unordered.

**Example**: $\{a,b\} \rightarrow \{f\}$
2) Partially-Ordered Sequential rules

**Partially-Ordered Sequential rules**: Rules of the form $X \rightarrow Y$, where $X$ and $Y$ are itemsets that are unordered.

**Example**: $\{a,b\} \rightarrow \{f\}$

**Interpretation**: If we observe $a$ and $f$ (in any order), they will be followed by $f$. 
2) Partially-Ordered Sequential rules

• This type of rule is often more interesting because it can summarize many standard sequential rules.

• For example:

\[
\{ \text{Vivaldi}, \{ \text{Mozart} \}, \{ \text{Handel} \} \} \Rightarrow \{ \text{Berlioz} \}, \\
\{ \text{Mozart}, \{ \text{Vivaldi} \}, \{ \text{Handel} \} \} \Rightarrow \{ \text{Berlioz} \}, \\
\{ \text{Handel}, \{ \text{Vivaldi} \}, \{ \text{Mozart} \} \} \Rightarrow \{ \text{Berlioz} \}, \\
\{ \text{Handel}, \{ \text{Vivaldi} \}, \{ \text{Mozart} \} \} \Rightarrow \{ \text{Berlioz} \}, \\
\{ \text{Handel}, \{ \text{Vivaldi} \}, \{ \text{Mozart} \} \} \Rightarrow \{ \text{Berlioz} \}.
\]

\[
\{ \text{Mozart}, \{ \text{Vivaldi} \}, \{ \text{Handel} \} \} \Rightarrow \{ \text{Berlioz} \}
\]
2) Partially-Ordered Sequential rules

• A **partially-ordered sequential rule** $X \rightarrow Y$ is a relationship between two disjoint and non empty itemsets $X,Y$.

• A sequential rule $X \rightarrow Y$ has **two properties**:
  – **Support**: the number of sequences where $X$ occurs before $Y$, divided by the number of sequences.
  – **Confidence** the number of sequences where $X$ occurs before $Y$, divided by the number of sequences where $X$ occurs.

• **The task**: finding all **valid rules**, rules with a support and confidence not less than user-defined thresholds $minSup$ and $minConf$ (Fournier-Viger, 2010).
An example of Sequential Rule Mining

Let say that $minSup = 0.5$ and $minConf = 0.5$:

<table>
<thead>
<tr>
<th>ID</th>
<th>Sequences</th>
</tr>
</thead>
<tbody>
<tr>
<td>seq1</td>
<td>{a, b}, {c}, {f}, {g}, {e}</td>
</tr>
<tr>
<td>seq2</td>
<td>{a, d}, {c}, {b}, {a, b, e, f}</td>
</tr>
<tr>
<td>seq3</td>
<td>{a}, {b}, {f}, {e}</td>
</tr>
<tr>
<td>seq4</td>
<td>{b}, {f, g}</td>
</tr>
</tbody>
</table>

A sequence database

<table>
<thead>
<tr>
<th>ID</th>
<th>Rule</th>
<th>Support</th>
<th>Confidence</th>
</tr>
</thead>
<tbody>
<tr>
<td>r1</td>
<td>{a, b, c} $\Rightarrow$ {e}</td>
<td>0.5</td>
<td>1.0</td>
</tr>
<tr>
<td>r2</td>
<td>{a} $\Rightarrow$ {c, e, f}</td>
<td>0.5</td>
<td>0.66</td>
</tr>
<tr>
<td>r3</td>
<td>{a, b} $\Rightarrow$ {e, f}</td>
<td>0.75</td>
<td>1.0</td>
</tr>
<tr>
<td>r4</td>
<td>{b} $\Rightarrow$ {e, f}</td>
<td>0.75</td>
<td>0.75</td>
</tr>
<tr>
<td>r5</td>
<td>{a} $\Rightarrow$ {e, f}</td>
<td>0.75</td>
<td>1.0</td>
</tr>
<tr>
<td>r6</td>
<td>{c} $\Rightarrow$ {f}</td>
<td>0.5</td>
<td>1.0</td>
</tr>
<tr>
<td>r7</td>
<td>{a} $\Rightarrow$ {b}</td>
<td>0.5</td>
<td>0.66</td>
</tr>
</tbody>
</table>

Some rules found
Several algorithms

- **CMRules, RuleGrowth, ERMiner**: find all the sequential rules
- **TRuleGrowth**: find sequential rules with a window constraint
- **TopSeqRules**: find the top-k sequential rules
- **TNS**: find top-k non-redundant sequential rules
- **HUSRM**: find high utility sequential rules
- ...

These algorithms directly find the rules!
Some applications

E-learning


Some applications

Manufacturing simulation


Quality control

Some applications

**Web page prefetching**


**Anti-pattern detection in service based systems,**


**Embedded systems**

Some applications

**Alarm sequence analysis**


**Recommendation**

Some applications

Restaurant recommendation

Customer behavior analysis
Conclusion

• Today, I have introduced **sequential rule mining**.

• An important topic in pattern mining.

• Sometimes also called **temporal association rule mining** or **episode rules**.

• There are also other variations.

• Source code and dataset in the **SPMF library**
Running an algorithm

Choose an algorithm: CM-SPAM
Choose input file: snake_192_converted.txt
Set output file: test.txt
Choose minsup (%): 0.96 (e.g. 0.5 or 50%)
Min pattern length (optional): 4 (e.g. 1 item)
Max pattern length (optional): (e.g. 10 items)
Max gap (optional): (e.g. 1, 2, 3)
Required items (optional): (e.g. 1, 2, 3)
Show sequence ids? (optional): (default: false)
Open output file: using SPMF viewer

Algorithm is running...
============ CM-SPAM v0.97 : STATISTICS =============
Total time ~ 135 ms
Frequent sequences count: 447
Max memory (mb): 39.53382110595703447
minsup 157
Intersection count 2141

http://www.philippe-fournier-viger.com/spmf/